Use of IP Addresses for High Rate Flooding Attack Detection

Ejaz Ahmed[1], George Mohay[1], Alan Tickle[1], Sajal Bhatia[1]

[1] Queensland University of Technology, GPO Box 2434
Brisbane, QLD 4001, Australia
{e.ahmed, g.mohay, ab.tickle, s.bhatia}@qut.edu.au

Abstract. High-rate flooding attacks (aka Distributed Denial of Service or DDoS attacks) continue to constitute a pernicious threat within the Internet domain. In this work we demonstrate how using packet source IP addresses coupled with a change-point analysis of the rate of arrival of new IP addresses may be sufficient to detect the onset of a high-rate flooding attack. Importantly, minimizing the number of features to be examined, directly addresses the issue of scalability of the detection process to higher network speeds. Using a proof of concept implementation we have shown how pre-onset IP addresses can be efficiently represented using a bit vector and used to modify a "white list" filter in a firewall as part of the mitigation strategy.

Keywords: IP addresses, bit vector, bloom filter, cumulative sum.

1. Introduction

It is now almost ten years since the first full-scale high-rate flooding attacks were unleashed on the Internet community [1]. The vector for those attacks was a set of compromised computer systems aka "zombies" that directed a high-volume stream of packets towards the target hosts. Ten years on, and notwithstanding their conceptual simplicity, high-rate flooding attacks (aka Distributed Denial of Service or DDoS attacks) using the same basic *modus operandi* still constitute an extremely pernicious threat within the Internet domain [2]. They also remain an unsolved problem. Consequently techniques and processes that detect and mitigate the impact of this form of attack continue to be an important and active area of research and development [3]. They also form the basis for a joint research project currently being undertaken under the auspices of the Australian and Indian Governments as part of the Australia-India Strategic Research Fund (AISRF). A key deliverable of this project is a DDoS Mitigation Module (DMM) i.e. a "network flooding attack mitigation tool" which integrates the detection and mitigation capability into a single device.

Two of the key challenges in implementing a workable solution to the DDoS problem are to operate the detection process at so-called "wire speed" i.e. speeds of at least 10Gbps [2, 4] and be able to activate a response in real-time to mitigate the impact of the attack in order to protect some downstream target which may be an application server or security device or a complete subnet. One important line of research is to utilize IP-address related features as the primary means of detecting an attack [5-10]. In this work we demonstrate how using packet source IP addresses coupled with a change-point analysis of the rate of arrival of new IP addresses may be sufficient to detect the onset of a high-rate flooding attack. Importantly, minimizing the number of features to be examined, directly addresses the issue of scalability of the detection process to higher network speeds. We also show how this information about potentially anomalous IP addresses could be used to modify a "white list" filter within a firewall dynamically as part of the mitigation strategy. We intend in future work to deploy and evaluate the approach with respect to protecting a security device, specifically an application firewall, ModSecurity.

The remainder of the paper is structured as follows. Section 2 reviews previous work on using features and characteristics of IP-addresses in detecting high-rate flooding attacks. Section 3 introduces the proposed DDoS Mitigation Module (DMM). It discusses the overall architecture of the DMM. Section 4 provides a detailed description of the NSP algorithm which classifies IP addresses based on their previous association with the host site and which is a core component in the DMM. Because performance of the NSP algorithm is crucial to its successful deployment actual operational networks, this section also shows a possible implementation based on a Bit-Vector approach. Section 5 shows the results of the experiments to test the "proof-of-concept" of the key ideas. Finally Section 6 summarizes the paper and outlines the future work.

2. Related Work

IP addresses play a pivotal role in identifying the communicating parties within the TCP/ IP suite of protocols. Consequently there is now a distinct body of knowledge surrounding the use of source IP address monitoring for detecting high-rate flooding attacks [6, 8-14] and in particular those in which the source IP-address had been "spoofed" [9]. Moreover such techniques are also capable of distinguishing anomalous network traffic (e.g. a DDoS attack) from a legitimate network event such as a so-called Flash Event [9, 11]. In addition IP address monitoring is able to detect attacks where each compromised host in the so-called "botnet herd" mimics the behavior of a legitimate user thereby making it difficult to distinguish between the normal network traffic and the attack traffic [3, 8, 10].

Central to any detection process is the requirement to identify a feature or set of features that appear more frequently in the target class but are less prominent in normal traffic and which capture the inherent features of an attack [3]. Also, in light of the adaptive behavior of the botnet masters, it is important to use features that are difficult or impossible for an attacker to change [3]. Over time various IP-address related features have been used in the detection process including basic features such

as the traffic volume per IP address [12]. Other features include the change in the number of network flows (i.e. distinct source/ destination IP address and port pairs) [11] as well as a change in the number of clients and in the pattern or distribution of clients across ISPs and networks [9]. The proportion of new source IP addresses seen by the target [3, 6, 8] have also been used along with features such as evidence of abrupt changes in traffic volume, flow dissymmetry as well as changes in the distribution of source IP addresses and the level of concentration of target IP addresses.

Two key issues in effectively utilizing IP address information in the detection process are the impracticality of storing statistical data for each of the 2^{32} elements in the IPv4 address space and the need accommodate a sharp increase in the rate of arrival of new source IP addresses during a DDoS attack [3]. This necessitates minimizing the number of IP addresses being tracked and optimizing the way in which information about IP addresses is stored. For example Gil and Poletto [12] used a dynamic 4-level 256-ary tree (MULTOPS) to collect traffic date for each IP address. However Peng et al. note that such a structure could itself succumb to a memory exhausting attack. Hence Peng et al. and Takada et al. [3, 6, 13] only store information on IP addresses that complete the TCP connect sequence correctly or send more than a threshold minimum number of packets. Peng et al. [8] also use a data structure comprising 2^{10} counters to (partially) aggregate the information about distinct IP addresses. Similarly Cheng et al. [10] isolate those IP addresses that are "new" and which are concentrated on a particular target IP address. Other authors have sought to exploit any inherent clustering of IP addresses. For example, Le [14] aggregated IP addresses using various subnet masks on the premise that in a Flash crowd most of the source IP addresses are close to each other whilst in a DDoS attack the sources are assumed to be more widely distributed. Finally, there is also a scalability issue for solutions in the IPv6 address space where the number of addresses is 2^{128}.

Once the feature vector has been created the remaining problem is to decide on a suitable algorithm that can use this information for detecting a DDoS attack. For example Cheng et al. [10] use a Support Vector Machine (SVM) classifier whilst Peng et al. [6] used the Cumulative Sum (CUSUM) algorithm and a computation involving a simplified Mahalanobis distance [8] to detect any abrupt change in the fraction of new IP addresses (on the basis that an abrupt change of the proportion of new source IP addresses is a strong indication of a DDoS attack). Takada et al. [13] also used the CSUM algorithm whereas Le et al. [14] used filtering techniques from the area of signal processing.

One of the earliest attempts at constructing a real-time adaptive system conceptually similar to the proposed DMM can be found in the work of Lee et al.[15-17]. Whilst their focus was on building an Intrusion Detection System, they demonstrated the importance of carefully segmenting and parallelizing the individual tasks to be performed in order to have a device that is capable of being scaled to operate in real-time (or close to real-time) on high-speed, high-volume network traffic. They also addressed the problem of updating the device's internal knowledge base once it has identified that a new attack has occurred. The basic idea used by Lee et al. is that of incremental learning i.e. adding to the existing knowledge base without having to completely retrain the underlying classifiers. Another example of previous work on adaptive IDS and one that appears to be more closely related to the objectives of this

project is that of the Cannady [18]. In particular Cannady examined the specific problem of adapting an IDS in dealing with DoS attacks. He also utilized feedback from the protected system in the form of system state variables such as CPU load, available memory, etc.
More recent work on adaptive IDS can be found in Xu [19] and Moosa et al. [20] (who specifically focused on adaptive application aware firewalls). All extend the range of algorithms that could potentially be used as classifiers within the proposed DMM (e.g. Fuzzy Logic, Support Vector Machines, etc.). Notwithstanding the many variations on the theme, it would appear that the basic incremental learning architecture proposed by Lee et al. [15-17] together with the ideas of Cannady [18] offer a good starting point for building the proposed DMM.

3. Detection Architecture

The design goal is for the DMM to have the capability to protect two "common monitoring environments" (i.e. security devices) viz. an application-aware firewall and a network-based intrusion detection system (IDS), from high-rate network flooding attacks. This includes protecting the devices under the so-called zero-day attack scenario i.e. a pattern of attack that has not previously appeared. Figure 1 shows a schematic of a version of the proposed DMM that would operate in real-time but off-line with the task of protecting an application-aware firewall (or an Intrusion Prevention System).

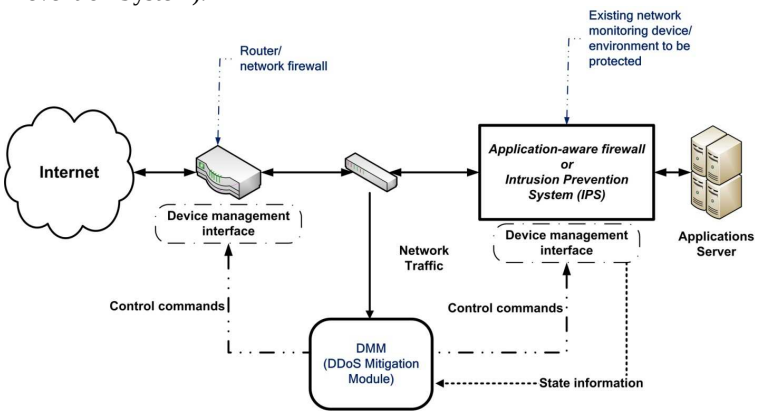

Fig. 1. Schematic of the proposed DMM protecting an application-aware firewall (or a Intrusion Prevention System).

The key functions of the proposed DDoS Mitigation Module (DMM) are to detect the onset of a DDoS attack, use "state information" about the set of devices to be protected by the DMM to predict if any of them are at imminent risk of failing, and formulate and then direct a set of "control commands" to the devices "at risk" to enable them to manage (or "shape") the network traffic situation (e.g. jettisoning

anomalous network packets). Since the intention is to develop a prototype DMM that can be deployed on existing networks, one of the initial design decisions is use COTS (Commercial off-the-shelf) offerings for components such as the designated "application-aware firewall" shown in Fig 1. As such, the designated "control commands" would then be rules expressed in the formats used internally in such devices. For example, one potential candidate "router/ firewall" device (at least for the purposes of the initial prototype) is a platform supporting the Open BSD Packet Filter (PF) [21]. (One of the key attractions of PF is that the filtering mechanism is externally configurable [22, 23].) In this case, the format of the designated "control commands" would be rules that conform to the PF syntax. Similarly the (Open Source Web) Application firewall ModSecurity [24, 25] is a potential candidate application firewall. (As with Packet Filter (PF) discussed above, ModSecurity has the underlying framework to deal with DoS attacks as well as an Automated Rule Update Capability.) As was the case with PF, the function of the DMM would be to generate rules in a format compatible with the ModSecurity rule set. A review of the literature highlights a degree of overlap between the core ideas of the proposed DMM and both current and previous work in constructing Intrusion Detection Systems that are adaptive (i.e. the ability to update the system rule set dynamically) and proactive (i.e. the ability to discern that an intrusion attack is in progress and react before it actually reaches its final stage) [26]. In fact, building an IDS with the capabilities to be adaptive and/ or proactive has been the subject of research in various guises for almost twenty years so there is a significant body of work from which it may be possible to draw a number of ideas as to how to approach the problem of building the required DMM.

4. Detection Algorithm

The basis of our algorithm is essentially that we expect to be able to detect whether currently incoming network traffic represents an attack on our system or not. We represent the two system states: NA (not under attack) and A (under attack) respectively. The algorithm is based on two top level functions, *ipac* and *ddos*.

The *ddos* function (described below) analyses the rate of arrival of packets from previously unseen IP addresses and on that basis assigns the system state to A or NA. It is invoked at regular intervals, and based on the rate of arrival of packets from previously unseen IP addresses determines a state change form NA to A, or vice versa, from A to NA.

The *ipac* address classification function examines the IP addresses of incoming network traffic. It extracts the source IP address of each incoming packet and determines if the IP address is new or not seen previously - NSP. To do so, *ipac* maintains two data structures which represent the IP addresses of incoming packets: W for 'Whitelist', and R for 'Recent'.

The data structure *W* is initialized with the source IP addresses of a known attack-free traffic sample, the data structure R is initialized to empty at system start-up. The point of *R* is that as packets arrive from new IP addresses, we need to note those IP addresses in a temporary location – the structure R. Only once the *ddos* function has

determined that the system is not under attack can those addresses be copied to the whitelist – the structure W. This avoids polluting the whitelist W. If on the other hand, the *ddos* function determines that the system is under attack, then R is discarded. The system starts in state NA.
The *ipac(ip)* function examines the source address *ip* of an incoming packet and if the IP address has not been seen previously, updates the structure *R*:

```
if NOT ((ip IN W) OR (ip IN R)) then INC(newIP)
add ip to R
```

The *ddos* function is invoked at regular intervals, intervals of the order of 1 to 10 seconds. It has a simple purpose: it analyses the recent change in arrival rate of packets from NSP IP addresses. It uses the function *StateChange* which in turn uses a cumulative sum algorithm (CUSUM) to detect abrupt changes in that rate of arrival and determines system state changes. The *ddos* function does as follows:

```
if (in state NA) then
      if NOT (StateChange(NA)) then  //no state change
            add R to W
      else                           //state change to A
            state = A
            communicate W to the protected security
            device to use as a white list
            R= empty

if ((in state A) then
      if (StateChange(A)) then     //state change to NA
      state = NA
      communicate to the protected security device to
      stop using the white list.
```

There are also two obvious limitations in the above algorithm. The first is that when the system is deemed to be under attack, that new IP addresses are treated as malicious – this can give rise to false positives. Further work will extend the basis for rejection of packets beyond the simple property of NSP. The further work will attempt to identify malicious traffic using other features also, to be used in tandem with NSP, features like subnet source addresses, IP distribution, traffic volume, traffic volume per IP address, etc. The second limitation concerns what to do about the interval of time during which *ipac* and *ddos* functions are executing. We hope to keep these functions very small and efficient, so this may not be a problem. It will however pose the problem of incoming packets not being properly processed if the functions are indeed too slow. We now discuss the implementation of the above NSP algorithm using bit vectors.

4.1. Bit Vector Implementation

For IPv4, a bit vector can be used for the implementation of W and R in the above algorithm. Using a bit vector, requires 0.5 GB to represent the 2^{32} address space of IPv4. (In contrast, an IPv6 bit vector needs to represent 2^{128} addresses. We are currently investigating the use of Bloom filters for IPv6).
Implementation via bit vectors allows for a number of optimizations to the algorithm. The step 'add R to W' in the *ddos* function can be optimized by modifying the *ipac* function so that when setting the appropriate bit in R, it keeps a temporary copy of the *ip* value, or better still, a copy of the address of the bit being set. This optimization allows the 'add R to W' step to be implemented as a series of set bit operations rather than having to OR the entire bit vector R into W. The second optimization relates to the step 'R = empty'. This step is optimized in a similar fashion, by unsetting the relevant bits rather than zeroising the entire bit vector.

4.2. Change Detection

On the onset of malicious activity, it is expected that the statistical properties of the traffic parameters no longer remain constant, resulting in an abrupt change. These change points can be detected using sequential analysis methods such as the cumulative sum (CUSUM) change point detection algorithm. CUSUM is a sequential analysis technique which assumes that the mean value of the (suitably transformed) parameter under observation will change from negative to positive in the event of a change in its statistical properties. Detecting this requires knowledge of the data distribution both before and after a malicious event.
In real time network traffic analysis, estimating traffic distribution both before and after a malicious event is rather a difficult task if not impossible due to the lack of a complete model. In change detection, this problem can be solved using a non-parametric CUSUM method as described by Blazek et al.[27]. In this paper we have adopted a non-parametric sliding window CUSUM change detection technique proposed by Ahmed et.al. [28, 29] for the analysis of source IP addresses. For the detailed description of the CUSUM technique, the interested readers are referred to [28, 29]. We use the CUSUM technique to detect changes in the number of new source IP addresses being observed in the network traffic during a measurement interval.

5. Experimentation

In order to establish the use of source IP address as a key feature to detect flooding attack, a comparative analysis of two different traffic traces has been conducted. In one analysis Peng et. al. [7] compared daily Auckland data trace [30] with the previous two week's data traces to observe the persistence of the source IP addresses. We have carried out a similar analysis using network traffic collected from a

dedicated block of unused IP addresses (commonly known as a Darknet) and Auckland traffic trace has been carried out. In contrast to used address spaces where there are production hosts connected to the Internet, unused address spaces are routable Internet addresses which do not have any production host connected to Internet. Due to the absence of any production host the traffic observed on a darknet is by definition unsolicited and likely to be either opportunistic or malicious. Table 1 provides the comparative analysis of the percentage of source IP addresses appearing in the fortnight previous to the listed dates for both Auckland and darknet network traffic traces.

Table 1. Persistance of source IP addresses.

Auckland Trace		Darknet Trace	
Date	Percentage	Date	Percentage
2001-Mar-26	88.7%	2009-Dec-25	1.25%
2001-Mar-27	90.3%	2009-Dec-26	1.05%
2001-Mar-28	89.1%	2009-Dec-27	1.03%
2001-Mar-29	89.2%	2009-Dec-28	0.91%
2001-Mar-30	90.2%	2009-Dec-29	0.94%
2001-Mar-31	89.9%	2009-Dec-30	1.07%
2001-Apr-01	88.1%	2009-Dec-31	7.99%

From Table 1, it can be observed that high percentage of source IP addresses appear in the last fortnight in Auckland traffic, a normal behavior in network traffic. In contrast, very low percentages of source IP's reappear during malicious network traffic such as the one collected from the darknet.
In order to evaluate the performance of the proposed NSP algorithm described in Section 4, the test bed shown in Figure 2 was being used.

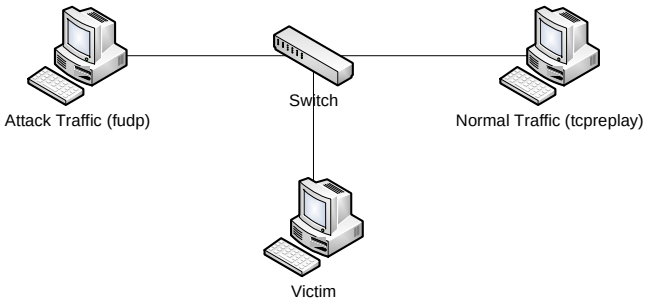

Fig. 2. The test bed architecture.

The normal network trace used in the experiment is the a real network traffic taken from the University of Auckland [30] known as Auckland VIII dataset. The IP addresses in the traffic trace have been mapped into 10.*.*.* using one to one hash mapping for privacy. The dataset is first analyzed to remove the SYN attacks which constitute the majority of the attacks [31]. In this regard TCP flows with less than 3

packets are treated as malicious packets and are being ignored. The cleaned data is then reproduced over the test bed using TCPREPLAY[1] utility. The traffic is replayed at the rate of around 300 packets per seconds. For attack traffic, fudp[2] utility was used to flood the victim machine with UDP packets with varying number of spoofed source IP addresses. The average attack traffic rate is set to around 3000 packets per second (10 times the normal background traffic). Table 2 provides the statistics of the attack traffic being used in the experimentation.

Table 2. Statistics of the attack traffic.

Unique Source IP	Number of Packets	Duration (seconds)	Traffic Rate(Mbps)
35	342141	134.38	1.01
40	328579	129.35	1.01
45	321433	129.29	1.01
50	313747	123.23	1.01
100	237831	93.49	1.01
150	201020	86.83	1.01

5.1. Performance Evaluation

In order to evaluate the performance of the proposed algorithm, the UDP flooding attack with varying number of source IP addresses has been embedded in the network traffic. UDP flooding with constant packet rate and constant set of source IP addresses has been generated, see Table 2 for detail. Due to space limitations, Figure 3 shows the result of UDP flood only for the case of 35 unique source IP addresses.

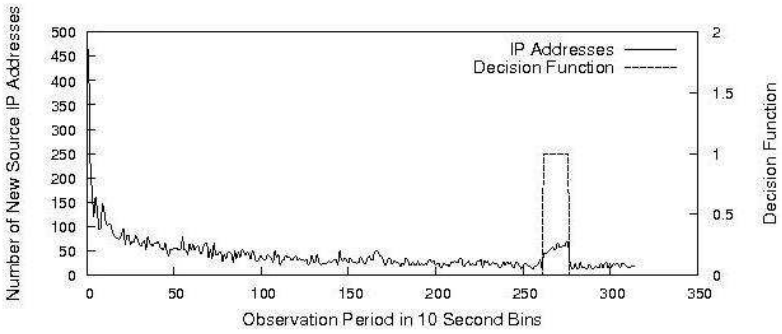

Fig. 3. UDP flooding attack with 35 source IP addresses.

[1] TCPReplay web page, http://tcpreplay.synfin.net/

[2] Fudp download page, http://linux.softpedia.com/get/Security/fudp-35626.shtm

In Figure 3 the horizontal axis represents the observation period in 10 second bins, the left vertical axis represents total number of new source IP addresses in the measurement interval and the right axis represents the CUSUM decision function with 1 being attack and 0 being no attack. The number of new source IP addresses in the 10 second measurement interval is calculated using the proposed algorithm described in Section 4. The UDP flooding attack was started at measurement interval 260 and ended at 280 and is detected by the CUSUM technique. The large increase in the new IP addresses at the start in Figure 3 is due to the fact that the bit vector *W* is empty at the start of the analysis. The subsequent inclusion of the source IP addresses in the bit vector *W* results in the constant decrease of new IP address counter. The CUSUM change detection technique is not applied during the training period, which is up to measurement interval 230, which enables the sliding window to learn the normal network traffic behavior. A manual analysis of the generated white list was performed post attack to check for the presence of any attack sources in the list. No attacking source IP addresses were found in the white list (for all the UDP attacks). The detection delay is bound by the measurement interval which is selected as 10 seconds in this paper. All the attacks listed in Table 2 have been detected in less than 10 seconds which can further be reduced using smaller measurement intervals.

6. Conclusion and Future Directions

In this paper, we have proposed a technique for detecting high rate flooding attacks. A proof of concept implementation of the proposed technique using bit vectors is provided in this paper. We have shown how a simple traffic feature such as source IP addresses can be used to effectively detect flooding attacks.

Our ongoing work focuses on implementing the algorithm using bloom filter in order to compare the results with the bit vector implementation and to enable us to extend the approach to IPv6 with its much bigger IP address space. In addition we seek to investigate the performance of both bit vectors and bloom filters under high speed network flooding attack in both IPv4 and IPv6 networks. Moreover the analysis of algorithm under different and diverse flooding attacks needs to be investigated.

We expect also in further work to extend the basis for rejection of packets beyond the simple property of IPs Not Seen Previously. The further work will attempt to identify malicious traffic using other features also, to be used in tandem with NSP, features like subnet source addresses, IP distribution, traffic volume, traffic volume per IP address, packet inter-arrival time etc.

Acknowledgement

This work was supported by the Australia-India Strategic Research Fund 2008-2011.

References

1. Garber, L., *Denial-of-Service Attacks Rip the Internet.* Computer, 2000. 33(4): p. 12-17.
2. Nazario, J., *Political DDoS: Estonia and Beyond (Invited Talk)*, in *17th USENIX Security Symposium*. 2008: San Jose, CA, USA.
3. Peng, T., C. Leckie, and K. Ramamohanarao, *Survey of network-based defense mechanisms countering the DoS and DDoS problems.* ACM Comput. Surv., 2007. 39(1): p. 3.
4. Miercom, *Enterprise Firewall: Lab Test Summary Report.* 2008.
5. Peng, T., C. Leckie, and K. Ramamohanarao, *Information sharing for distributed intrusion detection systems.* J. Netw. Comput. Appl., 2007. 30(3): p. 877-899.
6. Peng, T., C. Leckie, and K. Ramamohanarao, *Proactively Detecting Distributed Denial of Service Attacks Using Source IP Address Monitoring*, in *NETWORKING 2004, Networking Technologies, Services, and Protocols; Performance of Computer and Communication Networks; Mobile and Wireless Communications*. 2004. p. 771-782.
7. Peng, T., C. Leckie, and K. Ramamohanarao. *Protection from distributed denial of service attacks using history-based IP filtering.* in *Proceeding of the 38th IEEE International Conference on Communications (ICC 2003)*. 2003. Anchorage, Alaska.
8. Peng, T., C. Leckie, and K. Ramamohanarao, *System and Process For Detecting Anomalous Network Traffic*, W.I.P. Organisation, Editor. 2008.
9. Jung, J., B. Krishnamurthy, and M. Rabinovich. *Flash Crowds and Denial of Service Attacks: Characterization and Implications for CDNs and Web Sites*. in *Proceeding of 11th World Wide Web Conference*. 2002. Honolulu, Hawaii, USA.
10. Cheng, J., et al., *DDoS Attack Detection Algorithm Using IP Address Features* in *Frontiers in Algorithmics*. 2009, Springer Berlin / Heidelberg. p. 207-215.
11. Barford, P. and D. Plonka. *Characteristics of Network Traffic Flow Anomalies.* in *Proceedings of ACM SIGCOMM Internet Measurement Workshop* 2001.
12. Gil, T.M. and M. Poletto. *MULTOPS: A data-structure for bandwidth attack detection.* in *Proceedings of 10th Usenix Security Symposium*. 2001.
13. Takada, H.H. and A. Anzaloni. *Protecting servers against DDoS attacks with improved source IP address monitoring scheme*. in *2nd Conference on Next Generation Internet Design and Engineering (NGI '06)*. 2006.
14. Le, Q., M. Zhanikeev, and Y. Tanaka. *Methods of Distinguishing Flash Crowds from Spoofed DoS Attacks.* in *Next Generation Internet Networks, 3rd EuroNGI Conference on*. 2007.
15. Lee, W., S.J. Stolfo, and K.W. Mok, *Adaptive Intrusion Detection: A Data Mining Approach* Artificial Intelligence Review, 2000. 14(6): p. 533-567.
16. Lee, W. and S.J. Stolfo, *A framework for constructing features and models for intrusion detection systems.* ACM Trans. Inf. Syst. Secur., 2000. 3(4): p. 227-261.

17. Lee, W., et al. *Real time data mining-based intrusion detection*. in *DARPA Information Survivability Conference & Exposition II, 2001. DISCEX '01*. 2001. Anaheim, CA, USA.
18. Cannady, J. *Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks*. in *Proc. 23rd National Information Systems Security Conf., NISSC 2000*. 2000.
19. Xu, X., *Adaptive Intrusion Detection Based on Machine Learning: Feature Extraction, Classifier Construction and Sequential Pattern Prediction.* International Journal of Web Services Practices, 2006. 2(1-2): p. 49-58.
20. Moosa, A. and E.M. Alsaffar, *Proposing a hybrid-intelligent framework to secure e-government web applications*, in *Proceedings of the 2nd International Conference on Theory and Practice of Electronic Governance*. 2008, ACM: Cairo, Egypt.
21. OpenBSD. *PF: The OpenBSD Packet Filter*. 2009 [cited 11 November 2009]; Available from: http://www.openbsd.org/faq/pf/.
22. OpenBSD. *OpenBSD Programmer's Manual: ioctl - control device*. 2009; Available from: http://www.openbsd.org/cgi-bin/man.cgi?query=ioctl&sektion=2&arch=&apropos=0&manpath=OpenBSD+4.6.
23. OpenBSD. *OpenBSD System Manager's Manual: pfctl - control the packet filter (PF) device*. 2009 [cited 11 November 2009]; Available from: http://www.openbsd.org/cgi-bin/man.cgi?query=pfctl&sektion=8&arch=&apropos=0&manpath=OpenBSD+4.6.
24. Barnett, R. *ModSecurity Core Rule Set (CRS) v2.0*. 2009; Available from: http://www.owasp.org/index.php/File:OWASP_ModSecurity_Core_Rule_Set.ppt.
25. ModSecurity. *ModSecurity Open Source Web Application Firewall*. 2009; Available from: http://www.modsecurity.org/index.html.
26. Kabiri, P. and A.A. Ghorbani, *Research on Intrusion Detection and Response: A Survey*. International Journal of Network Security, 2005. 1(2): p. 84–102.
27. Tartakovsky, A.G., et al., *A novel approach to detection of intrusions in computer networks via adaptive sequential and batch-sequential change-point detection methods.* IEEE Transactions on Signal Processing, 2006. 54: p. 3372--3382.
28. Ahmed, E., A. Clark, and G. Mohay, *Change Detection in Large Repositories of Unsolicited Traffic*, in *The Fourth International Conference on Internet Monitoring and Protection (ICIMP 2009)*. 2009: Venice, Italy
29. Ahmed, E., A. Clark, and G. Mohay, *A Novel Sliding Window Based Change Detection Algorithm for Asymmetric Traffic*, in *IFIP International Conference on Network and Parallel Computing (NPC 2008)*. 2008: Shanghai, China. p. 168-175.
30. *Waikato Applied Network Dynamic Research Group.*, *http://wand.cs.waikato.ac.nz/*.
31. Mirkovic, J., et al. *DDoS Benchmarks and Experimenter's Workbench for the DETER Testbed.* in *3rd International Conference on Testbeds and Research Infrastructure for the Development of Networks and Communities (TridentCom 2007)* 2007.